FRAME IT: THE NUTS AND BOLTS OF RAD

Marty Brown, CommScope, Inc., Claremont, NC

INTRODUCTION

The road to finishing a quality application does not have to be a long and confusing one. With the advent of object oriented programming(OOP), that road has become much shorter and more direct. In addition to the benefits of the OOP approach, SAS/AF software has made programming the users' environment more straight forward by providing fundamental methods for several top level classes. Additionally, these methods are made available to each of their subclasses through inheritance. Methods necessary to provide functionality specific to each child class are provided at the child level. Methods needed to provide all of the routine functionality for all widgets are already included in the software. This fact alone reduces the time necessary to develop a quality application.

Application design is a step by step process that consists of seven distinct elements. The following are the steps essential to the development process:

+ **Concept Documentation**
+ **Module Design**
+ **Screen Design**
+ **Functionality**
+ **Debugging**
+ **Documentation**
+ **Release**

These steps, combined with the power of the SAS/AF development environment will ensure smooth travel to top quality applications in a short period of time.

The intent of this paper is to describe and discuss each step so that the reader can develop the methodology necessary to take a project from conception to release as smoothly and quickly as possible.

CONCEPT DOCUMENTATION

The first step to successfully developing an application is to determine exactly what the application will do. Surprisingly enough, this step in the development process is most often the one left out. It is impossible to develop an application without first knowing what the end result will be. This is the step that involves the most people. It should include all potential users, the developer, the project manager(if applicable), and, depending on the scope of the project, management.

Depending on how formal a company or the developer is, a functional specification can be written. A functional specification is an official document describing the project. It should include the nature of the project(e.g., data manipulation, data analysis, etc.), specific functionality, specific operating systems, screen designs, colors, etc. It could also specify cost limitations, and time constraints. A functional specification need not specify how to get from the beginning to the end; that is the job of the developer.

The developer should not begin work on the project until he/she is sure of the desired outcome. If no formal specification is required, the developer should note his/her ideas on paper and submit them to the appropriate personnel for their approval. Doing this will inevitably save time and avoid redoing work.

MODULE DESIGN

Once the purpose of an application is understood, it is then necessary to divide the project into small modules. By doing this the developer can organize himself/herself by breaking the entire project into small, attainable portions. This is the step where a rough draft of the layout of the application is designed. The draft can either be mental or a layout on paper of the functional modules or screen navigation. The actual layout of each screen need not be developed at this phase. Instead, the goal is to break the application into small pieces that are more readily programmed and are more easily worked with by the users.

For example, let's say that we have the following functional specification:

1) The application should accept input from users regarding criteria for subsetting a master data set.

2) The data must be subset by date as well as by one of two types.

3) It should run a program that obtains the subset data and then manipulates it to form a smaller data set for displaying the data in graphical form.

4) It should provide the users with the capability to see the specific data that made the graph only upon request from the user.

5) It should display the number of observations that made up the data.

6) The application must be accessed through the SAS toolbox.

From the specification, it seems that there are three logical modules. The first is a screen that accepts input from the user regarding the subsetting of the master data. Second, there is a screen that displays the data in a graphical format with functionality for retrieving the data based on user's request. Finally, there is a screen that displays the actual data from the users selection.

The modules are arranged in this way chiefly to make the flow of the program natural to the user as well as make programming more simple in that each module can be tested for its particular functionality without having the other modules in place. Obviously, there must be additional programming after the modules are finished in order for them to work together, but that is not a concern at this point.

Creating these modules is a great tool that aids in the visualization of the finished application. No longer is the developer creating screens, writing code, and testing, only to scrap it all and start over. Now that the modules have been decided upon, it is time for the next step.

SCREEN DESIGN

Aside from functionality, screen design is the most import aspect of the application from a user's perspective. The design of the screen(s) can have an impact on the success of the project. Screen design affects the users decision about the project through an intangible factor called perceived quality. Perceived quality can be low among the users even though all of the functionality that was requested is present and working. Therefore, it is at this step that at least some of the users become involved once again in the development process.

Users do not care about how an application is programmed, nor are they impressed by "slick" bits of code to provide the application's functionality. What they are interested in, however, is can they use it, is it comfortable, and does it work. Screen design focuses on the `can they use it` and `is it comfortable` aspects of the user's concerns.

There are several key factors to consider when developing screens. One is to know the users' mean level of computer experience. If the majority of users are computer novices, then the screens should be designed as straight forward and self explanatory as possible. If, however, most users are computer literate, then just providing good labels on widgets is probably enough. Another factor to consider is that of the operating system(s) that the users commonly use. Try to write the application on that operating system if possible. Also, understand that users form generalities about screen layouts of the operating system in which they are accustomed to using. People who "fly" through applications they use regularly do so because they know where things are, not because they read labels and messages. For instance, if the users are accustomed to using Microsoft Windows, then place widgets on the screen in the same locations as similar widgets on the operating system. Doing so will enhanced perceived quality immensely. Thirdly, make things as simple as possible. Do not make users go through ten menus to get the answer to a simple question. Instead, try to get the necessary information in two screens at most. Finally, do not crowd things into screens. It is better to design interfaces such that there is some white space on the screen but also be careful not to have too many empty areas.

After designing each of the screens, submit them to the users for approval. Keep in mind that at this point there is no functionality. The goal is to see if users understand the screens and to get feedback regarding the correctness of the modules or the functional specification. Once this is complete, incorporate any feasible suggestions from the users and move on to implementing functionality.

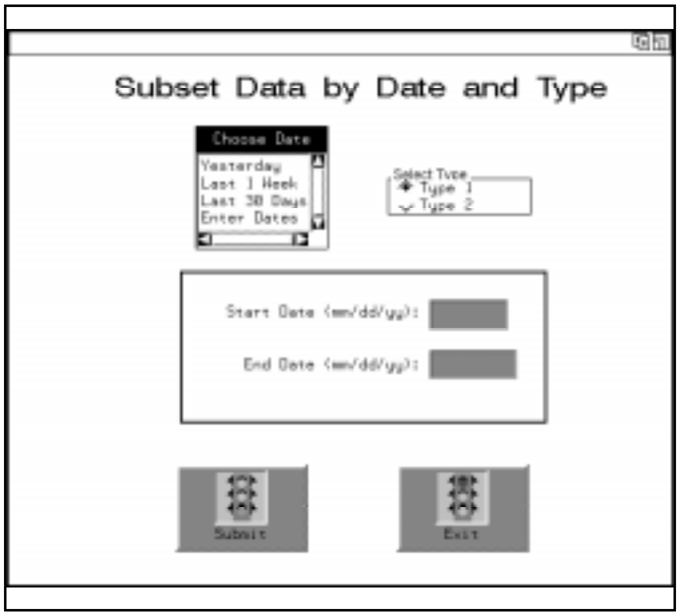

Figure 1A Screen One

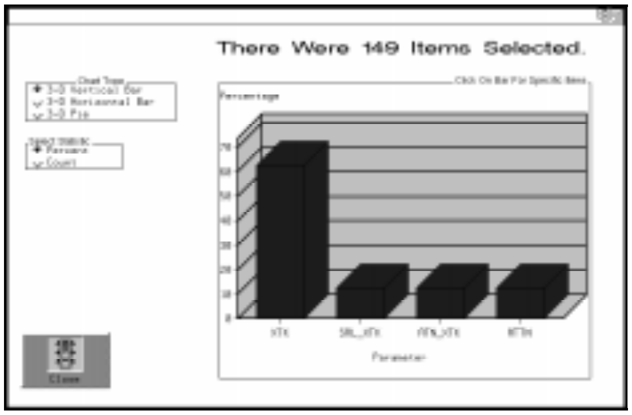

Figure 1B Screen Two

IMPLEMENT FUNCTIONALITY

After you and the users have settled on screen design, it is time to make the application do what the specification required. Since the application has already been broken into modules, it should only be a matter of writing some screen control language(SCL) to make the application work. Here again the goal is to make the modules work independently of each other unless it is absolutely necessary to make them work together. Since OOP can use widget names as labels in SCL, it is often not necessary to have a "main" section in the program. However, if no "main" section exists, then only the section of the program corresponding to name of the selected widget runs unless another labeled section is called through a link statement. If a "main" section does exist, it is executed every time any widget is selected.

Let us take the example mentioned earlier. The first module's program accepts input from the users and runs code that subsets and manipulates a master data set. A list box was put on the screen to make it simple for the users to select common dates. Also, a selection was included to let the user enter any date they wish. That selection is accompanied by a text entry field for the entering of the dates. The label shows the format that the user should use. A radio box gets the type selection from the user. Since a submit button is attached, there need not be any functionality to the program until the user clicks the button. The exception is the functionality provided through the attributes of the widgets. Once the submit button is activated, the program section with the same label executes the SCL required to subset the data. Depending on the type of manipulation of the data to be done, the data can be either subset through standard SCL functions or through the data step as executed through a submit block. Either way, the data must end up in a form that the graphics object on the next screen can use.

The next module calls for a screen to display the subset data in graphical form with the ability to retrieve specific data upon request. The specification did not declare what type of graph or what statistic the chart should display. Since SAS/AF provides for dynamic graph type and statistic selection, it was relatively simple to add these features via radio boxes while keeping the basic functionality in place. The SCL behind this screen basically has three labeled sections. One changes the graph type based on the selection on the first radio box. The second changes the chart statistic based on the selection of the second radio box. The last one tests to see if a bar chart is displayed and, if so, gets information about the bar that was activated, and further subsets the data for display on the next screen. This module also has a graphic text object to display the number of observations in the first subset of the master data set.

The final module contains a data table for displaying the detailed data based on the bar the user clicked on the previous screen. It receives information from the graphics screen and populates the data table. It also contains a command push button to return to the previous screen. The functionality of this module is minimal because it only has to display data that is passed to it through the call display routine from the graphics screen and the entry statement on the data table screen.

Once the functionality has been implemented for each module and the developer has tested it for obvious bugs, they should be submitted to at least some of the users for approval. The users should look for functionality, bugs, etc. They could also make suggestions for increased functionality if the developer has time and does not have to make major overhauls to accommodate them. Once the users approve the modules, it is time to put them together into a package and continue to the next step, debugging.

DEBUGGING

Debugging is an essential step in application development in that it is much better to discover potential problems before releasing an application to users than it is to patch an application that has already been distributed. Not only is it easier to do this, it maintains users/management's confidence in the developer to submit quality applications to the users.

In the debugging process, the developer should do everything in his/her power to try to make the application fail. This includes typing in dates in the wrong format, navigating screens contrary to the intended sequence, going back to start over, etc. Doing these and other things will surely find bugs if any exist. This step could involve some recoding or redesign. After completing this, the application should be sent back to the users for further testing. This cycle of debugging and testing should continue until the developer is satisfied that the application is error free.

DOCUMENTATION

The next step in the process is documentation. Documentation has two components. First, documentation of the code in order for the developer or other developers to understand it later. Also, documentation of the program and what it does. This could come in the form of a memo or in the form of an official report. The second component of documentation once again involves the users. The document should be reviewed by the users for clarity and completeness. It is this document that future users will use in order to be trained to use the application. Also, if a company is certified by an organization such as ISO9000, then this documentation may be required to comply with certification. Finally, it is time for the release of the application.

RELEASE

The final step in the application development life cycle is the release of the application. Once this step is complete, the only thing left is to maintain the program and perhaps train current and future users. Although this seems to be a trivial step, it can be difficult depending on how the users invoke the application. In the case with our example, the release is relatively simple. The functional specification called for the program to be accessed through the SAS toolbox. This is relatively easy in that the toolbox allows for editing by both system administrators as well as end users. Since users typically have their own toolbox defined in their own unique profile catalog, it would be more efficient to send a memo to all users of the new application describing the steps to take to modify their toolbox to be able to launch the new application. Whatever the method of release is used, once the users have the application, the last step of the development life cycle is complete.

SUMMARY

Application development is a step by step process. It has defined steps and small reachable goals. It is important to include all relevant personnel in the development throughout the process. After all, it is really the users that must be pleased with the released application.

Breaking the project into modules is key to eliminating guess work in application development. Once the modules have been set and the screens have been designed, it is time to get the users involved. Functionality is important, but too many options may get in the way when the goal of the application is to do a few relatively simple and routine tasks.

Testing and debugging will always benefit the developer in the long run in terms of reduced maintenance of the finished product. Once debugging is complete, documentation and release are all that are left to finishing the project.

The following is a flow chart of the life cycle of application development. It was created using the Process Flow Diagram available in release 6.11 of SAS/AF software.

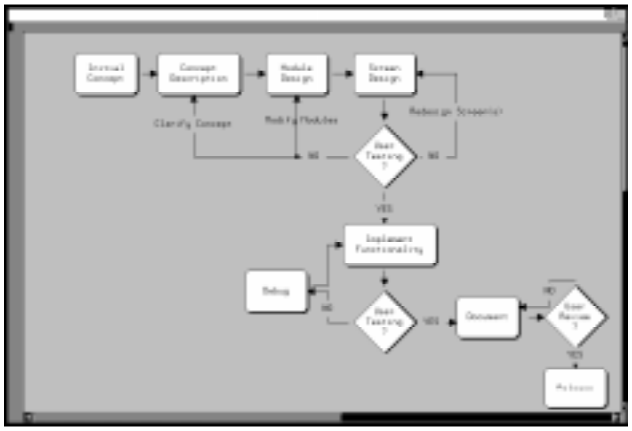

Figure 2 PDF Of Life Cycle

NOTES

SAS and SAS/AF are registered trademarks of SAS Institute, Inc., Cary, NC in the USA and other countries.